**Practical No. 13**

***Title:-***

|  |
| --- |
| Write a C program for Line Clipping using Cohen-Sutherland algorithm. |

***Relevant Course Outcome(s):-***

Implement Various Clipping algorithms and given curve generating algorithms.

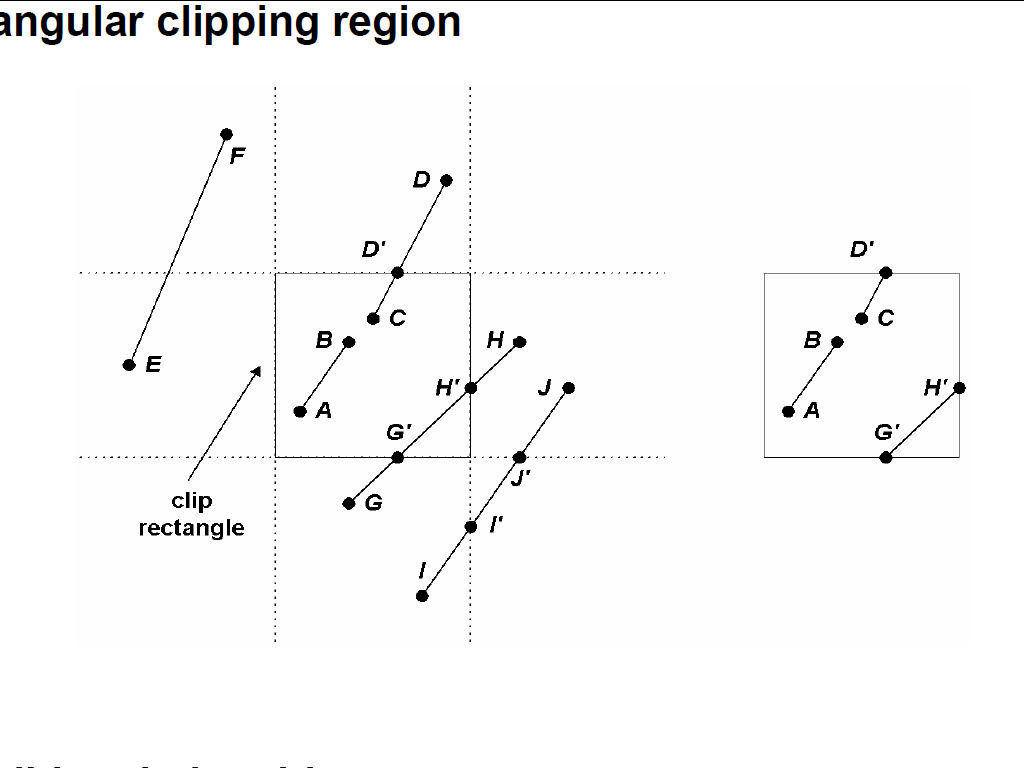
***Resources Required (Hardware & Softwares):-***

A Desktop PC/ Laptop

Ansi C/ Turbo C/ (Any distribution) installed

***Theory:-***

***Line Clipping Algorithm***



• Possible relationships between line positions and a standard rectangular clipping region

• Possible relationships Category 1 : Completely inside the clipping window

Category 2: Completely outside the window

Category 3: Partially inside the window

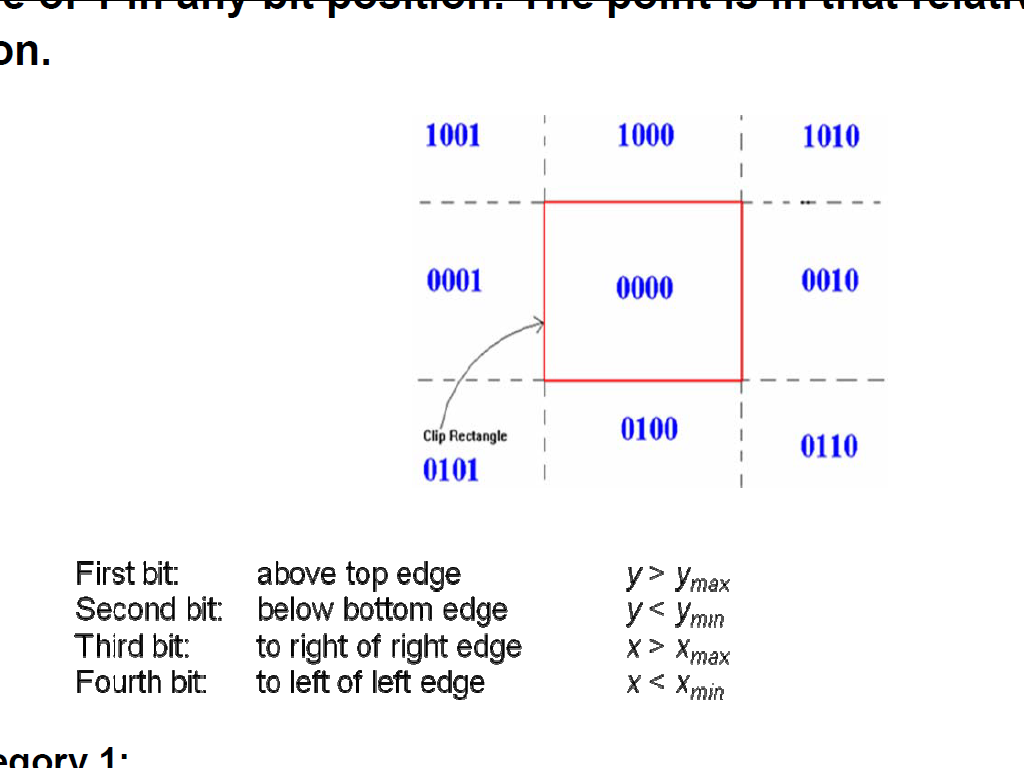
• Region code

• A four-digit binary code assigned to every line endpoint in a picture.

• Numbering the bit positions in the region code as 1 through 4 from right to left.

• Bit values in the region code determined by comparing endpoint coordinates to the clip boundaries

• A value of 1 in any bit position: The point is in that relative position.



**Category 1:**

The line is visible if region codes of both end points of line are 0000 i.e completely inside window. Hence accept line and display whole line

**Category 2:**

The line is not visible i.e line is completely outside of window, if bitwise logical AND of codes of both end points is not 0000 and thus reject the line and donot display

**Category 3:**

If bitwise logical AND of the region codes of both end points is 0000 then it is intersecting boundaries of clipping window.

1. To find intersection boundary:

– If bit 1 is 1 , it intersects with line y=ymax(top boundary)

– If bit 2 is 1 , it intersects with line y=ymin(bottom boundary)

– If bit 3 is 1 , it intersects with line x=xmax(right boundary)

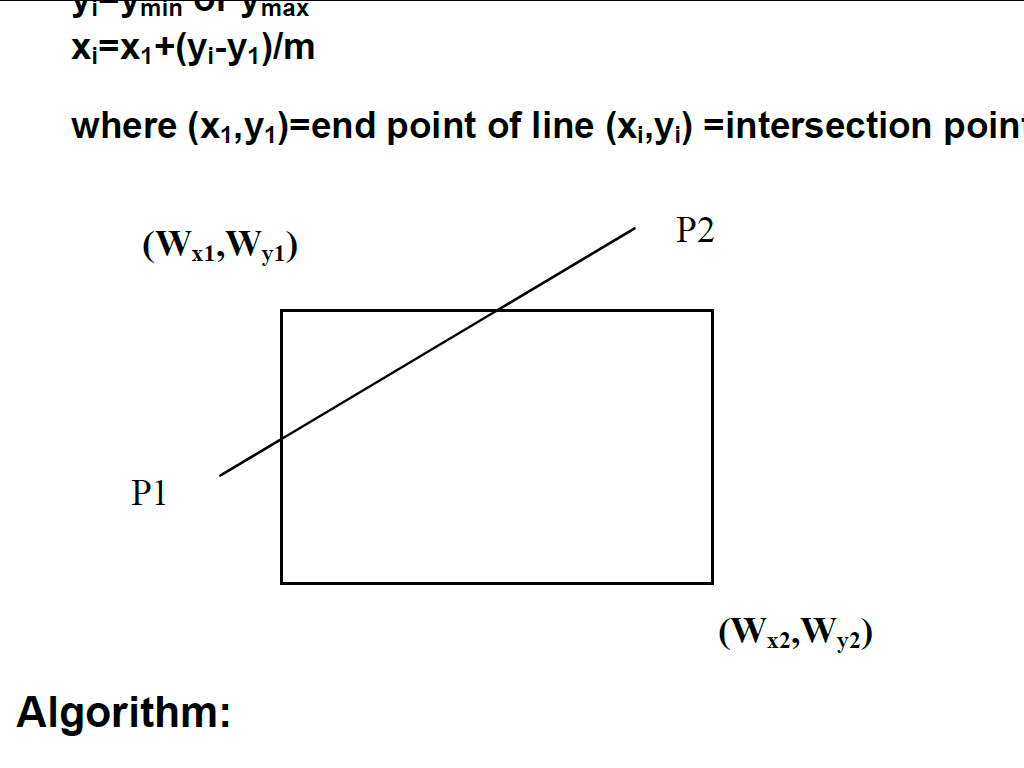
– If bit 4 is 1 , it intersects with line x=xmax(left boundary)

2. To find co-ordinates of intersection pt.:

– If boundary line is vertical then xi=xmin or xmax yi=y1+m(xi-x1)

– If boundary line is horizontal then yi=ymin or ymax xi=x1+(yi-y1)/m

where (x1,y1)=end point of line (xi,yi) =intersection point



***Algorithm:-***

1. Read two end points of the line say P1(x1,y1) and P2(x2,y2)

2. Read two corners (left-top and right bottom) of the window say(Wx1,Wy1 and Wx2,Wy2)

3. Assign the region codes for two endpoints P1 and p2 using following steps: Initialize code with bits 0000 Set Bit1-if(x<Wx1) Set Bit 2-if(x>Wx2)

Set Bit3-if(y<Wy2) Set Bit4-if(y>Wy1)

4. Check for visibility of line P1 P2:-

a) If region codes for both endpoints P1 and P2 are zero then the line is completely visible.

Hence draw the line goto step 9

b) If region codes for endpoints are not zero and the logical ANDing of them is also nonzero then the line is completely invisible so reject the line and goto step9

c) If region codes for two endpoints do not satisfy the conditions in 4a and 4b the line is partially visible.

5. Determine the intersecting edge of the clipping window by inspecting the region of end points

a) If region codes for both the end points are non-zero find the intersection points P1’ and P2’ with boundary edges of clipping window with respect to point P1 and point P2 resp.

b) If region code for any one end point is nonzero then find intersection points P1’ or P2’ with the boundary edge of the clipping window with respect to it.

6. Divide the line segment considering intersection points

7. Reject the line segment if any one end point of it appears outsides the clipping window.

8. Draw the remaining line segments

9. Stop.

**Program for Cohen Sutherland Line Clipping**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<dos.h>

#include<graphics.h>

#include<math.h>

typedef struct coordinate

{

int x,y;

char code[4];

}pt;

void drawwindow();

void drawline(pt p1,pt p2,int c1);

pt setcode(pt p);

int visibility(pt p1,pt p2);

pt resetendpt(pt p1,pt p2);

main()

{

int gd=DETECT,gm,v;

pt p1,p2,ptemp;

initgraph(&gd,&gm,"c:\\tc\\bgi");

cleardevice();

printf("\n\n\t\tEnter end pt.1(x,y): ");

scanf("%d %d",&p1.x,&p1.y);

printf("\n\n\t\tEnter end pt.2(x,y): ");

scanf("%d %d",&p2.x,&p2.y);

cleardevice();

drawwindow();

getch();

drawline(p1,p2,15);

getch();

p1=setcode(p1);

p2=setcode(p2);

v=visibility(p1,p2);

switch(v)

{

case 0:cleardevice();

drawwindow();

drawline(p1,p2,15);

break;

case 1:cleardevice();

drawwindow();

break;

case 2:cleardevice();

p1=resetendpt (p1,p2);

p2=resetendpt(p2,p1);

drawwindow();

drawline(p1,p2,15);

break;

}

getch();

closegraph();

return (0);

}

void drawwindow()

{

setcolor(WHITE);

line(150,100,450,100);

line(450,100,450,350);

line(450,350,150,350);

line(150,350,150,100);

}

void drawline(pt p1,pt p2,int c1)

{

setcolor(c1);

line(p1.x,p1.y,p2.x,p2.y);

}

pt setcode(pt p)

{

pt ptemp;

if(p.y<100)

ptemp.code[0]='1';

else

ptemp.code[0]='0';

if(p.y>350)

ptemp.code[1]='1';

else

ptemp.code[1]='0';

if(p.x>450)

ptemp.code[2]='1';

else

ptemp.code[2]='0';

if(p.x<150)

ptemp.code[3]='1';

else

ptemp.code[3]='0';

ptemp.x=p.x;

ptemp.y=p.y;

return(ptemp);

}

int visibility(pt p1,pt p2)

{

int i,flag=0;

for(i=0;i<4;i++)

{

if((p1.code[i]!='0')||(p2.code[i]!='0'))

flag=1;

}

if(flag==0)

return(0);

for(i=0;i<4;i++)

{

if((p1.code[i]==p2.code[i])&&(p1.code[i]=='1'))

flag=0;

}

if(flag==0)

return(1);

return(2);

}

pt resetendpt(pt p1,pt p2)

{

pt temp;

int x,y,i;

float m,k;

if(p1.code[3]=='1')

x=150;

if(p1.code[2]=='1')

x=450;

if((p1.code[3]=='1')||(p1.code[2]=='1'))

{

m=(float) (p2.y-p1.y)/(p2.x-p1.x);

k=(p1.y+(m\*(x-p1.x)));

temp.y=k;

temp.x=x;

for(i=0;i<4;i++)

//{

temp.code[i]=p1.code[i];

if((temp.y<=350)&&(temp.y>=100))

return(temp);

//}

}

if(p1.code[0]=='1')

y=100;

if(p1.code[1]=='1')

y=350;

if((p1.code[0]=='1')||(p1.code[1]=='1'))

{

m=(float)(p2.y-p1.y)/(p2.x-p1.x);

k=(float)p1.x+(float)(y-p1.y)/m;

temp.x=k;

temp.y=y;

for(i=0;i<4;i++)

//{

temp.code[i]=p1.code[i];

return(temp);

// }

}

else

return(p1);

}

**Output:- ( Paste your own Output )**

***Conclusion:-***

Thus, we have Implemented a C program for Line Clipping using Cohen-Sutherland algorithm.